

日本国特許庁
JAPAN PATENT OFFICE

50006-138
Atsushi SAKAI, del
10p76,625
Feb. 19, 2002
McDermott, Will &
Smyth

別紙添付の書類に記載されている事項は下記の出願書類に記載されて
いる事項と同一であることを証明する。

This is to certify that the annexed is a true copy of the following application as filed
with this Office

出願年月日
Date of Application: 2001年 2月21日

出願番号
Application Number: 特願2001-045072
[ST.10/C]: [JP2001-045072]

出願人
Applicant(s): 株式会社半導体理工学研究センター

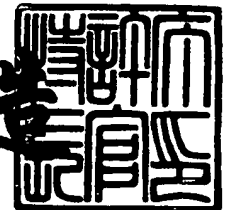
RECEIVED
MAY 14 2002
TC 2800 MAIL ROOM

CERTIFIED COPY OF
PRIORITY DOCUMENT

2002年 2月19日

特許庁長官
Commissioner,
Japan Patent Office

及川耕造



出証番号 出証特2002-3008619

【書類名】 特許願

【整理番号】 172782

【提出日】 平成13年 2月21日

【あて先】 特許庁長官殿

【国際特許分類】 G06F 12/08 310
G06F 15/16 645

【発明者】

 【住所又は居所】 東京都港区赤坂 7 - 5 - 5 1 富山県赤坂会館

 【氏名】 酒井 敦

【発明者】

 【住所又は居所】 神奈川県横浜市港北区日吉 7 - 1 6 - 2 7 - 1 0 3

 【氏名】 天野 英晴

【特許出願人】

 【識別番号】 396023993

 【住所又は居所】 東京都港区新橋 6 丁目 1 6 番 1 0 号

 【氏名又は名称】 株式会社半導体理工学研究センター

【代理人】

 【識別番号】 100062144

 【弁理士】

 【氏名又は名称】 青山 葆

【選任した代理人】

 【識別番号】 100086405

 【弁理士】

 【氏名又は名称】 河宮 治

【手数料の表示】

 【予納台帳番号】 013262

 【納付金額】 21,000円

【提出物件の目録】

 【物件名】 明細書 1

【物件名】 図面 1

【物件名】 要約書 1

【包括委任状番号】 9608010

【プルーフの要否】 要

【書類名】 明細書

【発明の名称】 キャッシュメモリシステム装置

【特許請求の範囲】

【請求項1】 プロセッサとメインメモリとの間に高速アクセスが可能な小容量のキャッシュメモリを設けてなるキャッシュメモリシステム装置において、

あらかじめプログラムされたソフトウェアにしたがって上記キャッシュメモリに対するデータ転送制御であるソフトウェア制御を行うソフトキャッシュ制御部と、

所定のハードウェアを用いて上記キャッシュメモリに対するデータ転送制御であるハードウェア制御を行うハードキャッシュ制御部と、

を備え、

上記プロセッサは、ソフトキャッシュ制御部に対してソフトウェア制御を実行させ、該ソフトウェア制御が実行不可能になると、ハードキャッシュ制御部に対してハードウェア制御を行わせることを特徴とするキャッシュメモリシステム装置。

【請求項2】 上記プロセッサは、ソフトウェア制御時にキャッシュミスが発生すると、上記ハードキャッシュ制御部に対してハードウェア制御を行わせることを特徴とする請求項1記載のキャッシュメモリシステム装置。

【請求項3】 上記ソフトキャッシュ制御部は、コンパイラの静的予測によって生成されたコードにしたがって、所望のデータをキャッシュメモリ内に格納することを特徴とする請求項1又は2記載のキャッシュメモリシステム装置。

【請求項4】 上記ソフトキャッシュ制御部は、メインメモリの所望のデータを読み出すデータ読み出し命令をプロセッサが実行する前に、該データ読み出し命令で指定されるメインメモリのアドレスのデータを読み出してキャッシュメモリに格納することを特徴とする請求項3記載のキャッシュメモリシステム装置。

【請求項5】 上記ソフトキャッシュ制御部は、メインメモリの所望のデータを読み出すデータ読み出し命令をプロセッサが実行すると同時に該データ読み出し命令で指定されたメインメモリのアドレスのデータをキャッシュメモリから

プロセッサに転送させることを特徴とする請求項4記載のキャッシュメモリシステム装置。

【請求項6】 上記ソフトキャッシュ制御部は、メインメモリにデータを書き込むデータ書き込み命令をプロセッサが実行する前に、キャッシュメモリに対してプロセッサからのデータを格納するアドレスを指定することを特徴とする請求項3、4又は5記載のキャッシュメモリシステム装置。

【請求項7】 上記ソフトキャッシュ制御部は、メインメモリにデータを書き込むデータ書き込み命令をプロセッサが実行し、指定したキャッシュメモリのアドレスに書き込まれたプロセッサからのデータを、該データ書き込み命令で指定されたメインメモリのアドレスに書き込むことを特徴とする請求項6記載のキャッシュメモリシステム装置。

【請求項8】 上記ハードキャッシュ制御部は、複数ウェイのセットアソシエイティブ方式を用いてキャッシュメモリのライン管理を行い、上記ソフトキャッシュ制御部は、該複数ウェイの内少なくとも1ウェイをフルアソシエイティブ方式を用いてキャッシュメモリのライン管理を行うことを特徴とする請求項1、2、3、4、5、6又は7記載のキャッシュメモリシステム装置。

【請求項9】 上記ソフトキャッシュ制御部は、キャッシュメモリに対するデータ転送制御を行う転送制御用プロセッサで構成されることを特徴とする請求項1、2、3、4、5、6、7又は8記載のキャッシュメモリシステム装置。

【発明の詳細な説明】

【0001】

【発明の属する技術分野】

本発明は、プロセッサとメインメモリとの間に高速アクセスが可能な小容量のキャッシュメモリを設けたキャッシュメモリシステム装置に関し、特に複数のプロセッサに対して無同期実行を行うマルチプロセッサシステムに使用するキャッシュメモリシステム装置に関する。

【0002】

【従来の技術】

図14は、従来のキャッシュメモリシステム装置の構成例を示した概略のプロ

ック図である。

図14において、キャッシュメモリシステム装置100は、プロセッサ101が、キャッシュメモリ部102を介してメインメモリ103に接続されて構成されている。更に、キャッシュメモリ部102は、タグメモリ105と、キャッシュメモリ106、タグメモリ105に格納されているタグの対応表を参照してキャッシュメモリ106に対するデータ転送制御を行うキャッシュ制御部107で構成されている。

【0003】

一方、キャッシュメモリシステム装置100では、キャッシュメモリ106にデータがあるか否か（ヒットするか否か）によって、アクセス時間が変わる。このため、キャッシュ制御部107は、キャッシュヒット率を向上させるために、あらかじめキャッシュメモリ106にデータを準備するプリフェッチ機構を有する場合がある。このようなキャッシュメモリシステム装置100において、プロセッサ101は、キャッシュメモリ106にアクセスするデータが存在する場合はキャッシュメモリ106から、キャッシュメモリ106にアクセスするデータが存在しない場合はメインメモリ103からデータが供給される。

【0004】

【発明が解決しようとする課題】

しかし、このような構成では、静的スケジューリング技術によってプリフェッチ機構を使用したとしても、キャッシュのヒット率を100%にすることは不可能であった。このため、マルチプロセッサ構成で無同期実行を行うマルチプロセッサシステム装置では、図14で示したような従来のキャッシュメモリシステム装置を使用して無同期実行を行うことは困難であった。

【0005】

本発明は、上記のような問題を解決するためになされたものであり、データ転送を管理するプロセッサを追加すると共にソフトウェアで制御される動作モードを追加して、コンパイラにキャッシュメモリのライン情報を管理する機構を設けることによって、キャッシュミスが発生せずに無同期実行が可能となるキャッシュメモリシステム装置を得ることを目的とする。

【0006】

【課題を解決するための手段】

この発明に係るキャッシュメモリシステム装置は、プロセッサとメインメモリとの間に高速アクセスが可能な小容量のキャッシュメモリを設けてなるキャッシュメモリシステム装置において、あらかじめプログラムされたソフトウェアに従ってキャッシュメモリに対するデータ転送制御であるソフトウェア制御を行うソフトキャッシュ制御部と、所定のハードウェアを用いてキャッシュメモリに対するデータ転送制御であるハードウェア制御を行うハードキャッシュ制御部とを備え、プロセッサは、ソフトキャッシュ制御部に対してソフトウェア制御を実行させ、該ソフトウェア制御が実行不可能になると、ハードキャッシュ制御部に対してハードウェア制御を行わせるものである。

【0007】

具体的には、上記プロセッサは、ソフトウェア制御時にキャッシュミスが発生すると、ハードキャッシュ制御部に対してハードウェア制御を行わせるようにした。

【0008】

また、上記ソフトキャッシュ制御部は、コンパイラの静的予測によって生成されたコードにしたがって、所望のデータをキャッシュメモリ内に格納するようにしてもよい。

【0009】

具体的には、上記ソフトキャッシュ制御部は、メインメモリの所望のデータを読み出すデータ読み出し命令をプロセッサが実行する前に、該データ読み出し命令で指定されるメインメモリのアドレスのデータを読み出してキャッシュメモリに格納するようにした。

【0010】

更に、上記ソフトキャッシュ制御部は、メインメモリの所望のデータを読み出すデータ読み出し命令をプロセッサが実行すると同時に該データ読み出し命令で指定されたメインメモリのアドレスのデータをキャッシュメモリからプロセッサに転送させるようにしてもよい。

【0011】

一方、上記ソフトキャッシュ制御部は、メインメモリにデータを書き込むデータ書き込み命令をプロセッサが実行する前に、キャッシュメモリに対してプロセッサからのデータを格納するアドレスを指定するようにした。

【0012】

更に、上記ソフトキャッシュ制御部は、メインメモリにデータを書き込むデータ書き込み命令をプロセッサが実行し、指定したキャッシュメモリのアドレスに書き込まれたプロセッサからのデータを、該データ書き込み命令で指定されたメインメモリのアドレスに書き込むようにしてもよい。

【0013】

また、上記ハードキャッシュ制御部は、複数ウェイのセットアソシエイティブ方式を用いてキャッシュメモリのライン管理を行い、上記ソフトキャッシュ制御部は、該複数ウェイの内少なくとも1ウェイをフルアソシエイティブ方式を用いてキャッシュメモリのライン管理を行うようにした。

【0014】

また具体的には、上記ソフトキャッシュ制御部は、キャッシュメモリに対するデータ転送制御を行う転送制御用プロセッサで構成されるようにした。

【0015】

【発明の実施の形態】

次に、図面に示す実施の形態に基づいて、本発明を詳細に説明する。

図1は、本発明の実施の形態におけるキャッシュメモリシステム装置を使用するマルチプロセッサシステムの例を示した概略のブロック図である。

図1において、マルチプロセッサシステムは、複数のプロセッサCPU1～CPU n (n は、 $n > 1$ の自然数)と、該各プロセッサCPU1～CPU n に対応して設けられたキャッシュメモリ部CM1～CM n と、集中共有メモリ (centralized shared memory) CSMと、データ交信用バス3とを備えている。各プロセッサCPU1～CPU n は、対応するキャッシュメモリ部CM1～CM n に接続され、各キャッシュメモリ部CM1～CM n は、データ交信用バス3を介して互いに接続されると共に集中共有メモリCSMに接続されている。

【 0 0 1 6 】

なお、このような構成において、各プロセッサCPU1～CPU_nの通信には、各プロセッサのレジスタ間で通信するプロセッサ間通信というシステムを使用しているが、通信データが多くレジスタ間通信では処理しきれない場合は、キャッシュメモリ部CM1～CM_nと集中共有メモリCSMとの間に、それぞれ通信用のローカルメモリCOMM1～COMM_nを対応させて設けるようにしてもよい。この場合、各プロセッサCPU1～CPU_nは、指定するアドレスによってローカルメモリCOMM1～COMM_n又は集中共有メモリCSMにアクセスすることができる。

【 0 0 1 7 】

図2は、本発明の実施の形態におけるキャッシュメモリシステム装置の例を示した概略のブロック図であり、図2では、図1で示した任意の1つのプロセッサCPU_m（ $m=1\sim n$ ）の場合を例にして示している。

図2において、キャッシュメモリシステム装置1は、プロセッサCPU_mと、キャッシュメモリ部CM_mと、メインメモリをなす集中共有メモリCSMとで構成されている。

【 0 0 1 8 】

また、キャッシュメモリ部CM_mは、リードバッファ5を有するキャッシュメモリ6と、タグメモリ7と、ソフトウェア制御時に該キャッシュメモリ6に対するデータ転送制御を行うDTC（Data Transfer Controller）8と、ハードウェア制御時に該キャッシュメモリ6に対するデータ転送制御を行うハードキャッシュ制御部9とで構成されている。DTC8は、ソフトウェア制御時にプロセッサCPU_mから入力されるアドレスデータのチェックを行うアドレスチェック機構を備え、キャッシュメモリ6に対するデータ転送の管理を行うプロセッサで構成され、ソフトキャッシュ制御部をなすものである。

【 0 0 1 9 】

集中共有メモリCSMのアドレスは、タグ（tag）部とキー（key）部に分けられ、タグメモリ7は、キャッシュメモリ6のどの場所に集中共有メモリCSMのどのアドレスのデータが格納されているかの対応を示すタグとキーの対応表を格

納する。ソフトウェア制御時にはDTC8が、ハードウェア制御時にはハードキャッシュ制御部9が、タグメモリ7に格納されている対応表のデータ管理を行う。このことから、DTC8及びハードキャッシュ制御部9は、タグメモリ7の対応表のタグをチェックすることによって、プロセッサCPUmから要求された集中共有メモリCSMのアドレスのデータがキャッシュメモリ6内にあるか否かが分かる。

【0020】

ハードキャッシュ制御部9は、プロセッサCPUmから要求された集中共有メモリCSMのアドレスのデータがキャッシュメモリ6内にあった場合、キャッシュメモリ6のどのアドレスに格納されているかがタグから分かるため、キャッシュメモリ6に該アドレスを指定してプロセッサCPUmに読み出すようにする。また、ハードキャッシュ制御部9は、プロセッサCPUmから要求された集中共有メモリCSMのアドレスのデータがキャッシュメモリ6内になかった場合、プロセッサCPUmから要求されたデータを集中共有メモリCSMからキャッシュメモリ6へ読み出す。このとき、ハードキャッシュ制御部9は、集中共有メモリCSMからのデータをキャッシュメモリ6のどのアドレスに格納するかを決定し、キャッシュメモリ6に該決定したアドレスを指定することによって、該アドレスに集中共有メモリCSMから読み出されたデータが格納される。

【0021】

ここで、ソフトウェア制御とは、コンパイラの静的予測によって生成されたコードにしたがってデータをキャッシュメモリ6上に準備する動作モードであり、ある一定区間内のキャッシュのヒット率を100%にして、マルチプロセッサ構成時に無同期実行を可能にする。これに対してハードウェア制御とは、従来行われている制御であり、プロセッサCPUmからの要求があつて初めてキャッシュメモリ6に対して、マッピング、キャッシュヒットの確認、キャッシュメモリ6からのデータの追い出し、及び集中共有メモリCSMからキャッシュメモリ6へのデータ転送が行われる動作モードである。

【0022】

なお、ハードウェア制御では、キャッシュメモリ6のどの場所にどのデータを

格納するか、どのデータをキャッシュメモリ 6 から追い出すか等はすべてハードウェアで管理されることになるが、今後の挙動を把握しているわけではない。このことから、過去のアクセスパターンを利用した何らかの法則に基づいて配置を決める。例えば、格納場所に関しては 4 ウェイ・セットアソシエイティブ (4-way set-associative) 等の方法で、データ追い出しの管理に関しては F I F O、L R U 等のハードウェアで実現できる範囲の方法で行う。

【 0 0 2 3 】

図 3 は、4 ウェイ・セットアソシエイティブ方式を用いた場合のタグメモリ 7 に格納されているタグの対応表の例を示した図であり、図 3 では、8 種類のタグを有する場合を例にして示している。

図 3 において、タグ 0 には 8 の倍数のキーが、タグ 1 には「8 の倍数 + 1」のキーが第 1 ～第 4 ウェイ (way) の各ウェイごとにそれぞれ対応付けされているように、タグ n ($n = 0 \sim 7$) には「8 の倍数 + n 」のキーが各ウェイ (way) ごとに対応付けされている。このように、タグに対するキーを何らかの法則に基づいて各ウェイごとに配置されており、ハードウェア制御時には、ハードキャッシュ制御部 9 は、このような 4 ウェイ・セットアソシエイティブ方式を用いてキャッシュメモリ 6 のライン管理を行う。

【 0 0 2 4 】

一方、DTC 8 は、プロセッサ CPU m とキャッシュメモリ 6 との間、及びキャッシュメモリ 6 と集中共有メモリ CSM との間のそれぞれのデータ転送を、コンパイラがあらかじめ生成した命令列にしたがって制御するプロセッサである。DTC 8 とコンパイラのデータ解析を組み合わせることで、所望のデータをあらかじめキャッシュメモリ 6 に用意して、キャッシュのヒット率を 100% にすることができる。更に、DTC 8 は、ソフトウェア制御時に、少なくとも 1 ウェイのみフルアソシエイティブ (full-associative) 方式を用い、他のウェイはセットアソシエイティブ方式を用いてキャッシュメモリ 6 のライン管理を行う。

【 0 0 2 5 】

図 4 は、図 3 で示したタグの対応表のソフトウェア制御時における例を示した図である。なお、図 4 では、4 ウェイの内、第 4 ウェイのみにフルアソシエイテ

イブ方式を用いる場合を例にして示している。

図4において、第1～第3ウェイは、ハードウェア制御時と同様にセットアソシエイティブ方式を用い、第4ウェイのみフルアソシエイティブ方式を用いてキャッシュメモリ6のライン管理が行われる。

【0026】

何らかの法則に基づいて配置を決めるセットアソシエイティブ方式に対して、フルアソシエイティブ方式では、使用されていないキャッシュメモリ6のアドレスに自由に配置することができる。このため、セットアソシエイティブ方式の領域でデータの衝突が発生し、データを排除しなければ新しいデータが読み込めないような場合、DTC8は、新しいデータをフルアソシエイティブ方式の領域の使用されていないアドレスに読み込むことによってキャッシュの利用効率を高めることができる。

【0027】

一方、ソフトウェア制御からハードウェア制御に移行した場合、図5で示すように、ハードキャッシュ制御部9によって、フルアソシエイティブ方式の領域でセットアソシエイティブ方式の法則に基づかないタグのキーは削除されて、タグの対応表の更新が行われる。

【0028】

次に、コンパイラによって生成されるDTC8への命令について説明する。DTC8への命令としては、キャッシュメモリ6とプロセッサCPUmとの仲介を行う際に発行される「LAC」, 「SAC」と、キャッシュメモリ6と集中共有メモリCSMとの仲介を行う際に発行される「PL (PreLoad)」, 「PS (PostStore)」がある。LAC命令は、キャッシュメモリ6の特定部分のデータをプロセッサCPUmに読み出させる命令であり、SAC命令は、キャッシュメモリ6の特定の部分にデータを書き込ませる命令である。また、PL命令は、集中共有メモリCSMからキャッシュメモリ6の特定部分にデータを読み出させる命令であり、PS命令は、キャッシュメモリ6の特定部分のデータを集中共有メモリCSMに書き込ませる命令である。

【0029】

ソフトウェア制御は、コンパイラによるデータアクセスの静的解析が完全に可能な部分で行われることから、該静的解析を生かしてコンパイル時にキャッシュメモリ6まで制御を行うことができ、プレロード (PreLoad) 等も行うことができる。このため、コンパイラがキャッシュメモリ6の制御を行った場合、当然キャッシュメモリ6のアドレスマップはコンパイラ内で管理される。したがって、実際の動作時にはハードウェアにアドレスマップがないため、プロセッサCPUmがロード (Load) 命令で集中共有メモリCSMのアドレスを指定しても、該アドレスのデータがキャッシュメモリ6のどの場所に格納されているかは不明となる。これを解決するために、ソフトウェア制御では、キャッシュメモリ6のアドレスを指定したLAC命令及びSAC命令を使用して、ロード及びストア (Store) を行う。

【0030】

DTC8が実行する命令のフォーマットとしては、「LAC cache-addr」、「SAC cache-addr」、「PL cache-addr mem-addr」及び「PS cache-addr mem-addr」となる。なお、「cache-addr」には、キャッシュメモリ6のアドレスが、「mem-addr」には、集中共有メモリCSMのアドレスが示される。「LAC cache-addr」は、「cache-addr」で示されたキャッシュメモリ6のアドレスのデータを読み出してリードバッファ5に一時的に格納させ、プロセッサCPUmにロード命令が発行されてから1クロックでプロセッサCPUmに出力できるようにする命令である。「LAC cache-addr」は、「cache-addr」に示されたキャッシュメモリ6のアドレスのデータを読み出すロード命令がプロセッサCPUmに発行される1クロック前に、DTC8に発行される。

【0031】

「SAC cache-addr」は、「cache-addr」で示されたキャッシュメモリ6のアドレスにプロセッサCPUmからのデータを格納させる命令である。「SAC cache-addr」は、「cache-addr」で示されたキャッシュメモリ6のアドレスにデータを書き込むストア命令がプロセッサCPUmに発行される1クロック前に、DTC8に発行される。

【0032】

「PL cache-addr mem-addr」は、「mem-addr」で示された集中共有メモリCSMのアドレスのデータを読み出して、「cache-addr」で示されたキャッシュメモリ6のアドレスに書き込ませる命令である。「PL cache-addr mem-addr」は、「cache-addr」に示されたキャッシュメモリ6のアドレスのデータを読み出すロード命令がプロセッサCPUmに発行されるよりも、少なくとも集中共有メモリCSMからデータを読み出してキャッシュメモリ6に書き込まれるのに要するクロック数だけ前にDTC8に発行される。

【0033】

「PS cache-addr mem-addr」は、「cache-addr」で示されたキャッシュメモリ6のアドレスのデータを読み出して、「mem-addr」で示された集中共有メモリCSMのアドレスに書き込ませる命令である。「PS cache-addr mem-addr」は、「cache-addr」に示されたキャッシュメモリ6のアドレスのデータを集中共有メモリCSMに書き込ませるストア命令がプロセッサCPUmに発行された直後、例えばプロセッサCPUmに該ストア命令が発行される1クロック後に、DTC8に発行される。

【0034】

このように、DTC8は、コンパイラによってスケジューリングされた命令にしたがって動作し、ソフトウェア制御では、キャッシュメモリ6内のどのアドレスにデータを格納するかというところまで管理を行う。このことから、意図しないキャッシュメモリ6内のデータの追い出しを防止することができ、不必要なデータの消去等ができることから、キャッシュのヒット率を100%にすることができる。更に、プロセッサCPUmの集中共有メモリCSMへのアクセスが高速になるだけでなく、集中共有メモリCSMへのアクセスが安定することによって、クロックレベルでのプロセッサCPUmの動作の予測が可能となり、マルチプロセッサシステム装置で使用した場合に、各プロセッサによる無同期実行等を実現することができる。

【0035】

次に、ソフトウェア制御時のプロセッサCPUm及びDTC8の動作について説明する。

図 6 は、プレロード実行時における各部の動作例を示したブロック図であり、図 6 を用いてプレロード実行時における各部の動作を説明する。なお、点線の矢印が制御の流れを、実線の矢印がデータの流れを示しており、ソフトウェア制御時の動作説明に関係しない部分は省略している。

【 0 0 3 6 】

図 6 において、プレロードを実行する場合、まず D T C 8 は、P L 命令が発行され、該 P L 命令で指定されたキャッシュメモリ 6 のアドレスに有効なデータが格納されていないことをチェックし、有効なデータが格納されている場合は、所定のミス信号 *miss* をプロセッサ C P U m に出力する。プロセッサ C P U m は、ミス信号 *miss* が入力されると、ソフトウェア制御からハードキャッシュ制御部 9 を使用するハードウェア制御に切り替える。また、有効なデータが格納されていない場合、D T C 8 は、該 P L 命令で指定された集中共有メモリ C S M のアドレスのデータを読み出して、該 P L 命令で指定されたキャッシュメモリ 6 のアドレスに格納させる。

【 0 0 3 7 】

次に、D T C 8 は、L A C 命令が発行され、該 L A C 命令で指定されたキャッシュメモリ 6 のアドレスのデータを読み出して、リードバッファ 5 に格納させる。次に、プロセッサ C P U m は、ロード命令が発行され、該ロード命令で指定された集中共有メモリ C S M のアドレスを D T C 8 に出力する。D T C 8 は、入力されたアドレスが上記 P L 命令で指定された集中共有メモリ C S M のアドレスと一致するか否かをチェックし、一致した場合は、キャッシュヒットしたことを示す所定のヒット信号 *hit* をプロセッサ C P U m 及びリードバッファ 5 にそれぞれ出力する。ヒット信号 *hit* が入力されたリードバッファ 5 は、格納しているデータをプロセッサ C P U m に出力する。このようにして、D T C 8 によるプレロードが行われる。

【 0 0 3 8 】

一方、D T C 8 は、入力されたアドレスが上記 P L 命令で指定された集中共有メモリ C S M のアドレスと一致しなかった場合は、キャッシュミスしたことを示す所定のミス信号 *miss* をプロセッサ C P U m に出力する。プロセッサ C P U

mは、ミス信号missが入力されると、ソフトウェア制御からハードキャッシュ制御部9を使用するハードウェア制御に切り替える。

【0039】

図7は、ハードウェア制御時においてロード命令を実行しキャッシュミスした場合の各部の動作例を示したブロック図である。

図7において、プロセッサCPU_mは、ロード命令が発行され、該ロード命令で指定された集中共有メモリCSMのアドレスをハードキャッシュ制御部9に出力する。ここで、タグメモリ7には、キャッシュメモリ6のどの場所に集中共有メモリCSMのどのアドレスのデータが格納されているかを示す対応表であるタグが格納されている。ハードキャッシュ制御部9は、ハードウェア制御時において該タグメモリ7に格納されているタグの内容を管理している。

【0040】

ハードキャッシュ制御部9は、タグをチェックしてプロセッサCPU_mから入力された集中共有メモリCSMのアドレスのデータがキャッシュメモリ6内にあるか否かをチェックする。キャッシュメモリ6内にプロセッサCPU_mから指定されたアドレスのデータがなかった場合、ハードキャッシュ制御部9は、キャッシュミスしたことを示す所定のミス信号missをプロセッサCPU_mに出力する。次に、ハードキャッシュ制御部9は、プロセッサCPU_mから指定されたアドレスのデータを集中共有メモリCSMから読み出してキャッシュメモリ6に格納してタグメモリ7内のタグの対応表を更新する。更に、ハードキャッシュ制御部9は、更新した対応表に基づいて、プロセッサCPU_mから指定されたアドレスのデータをキャッシュメモリ6から読み出してプロセッサCPU_mに出力する。

【0041】

一方、図8は、ハードウェア制御時においてロード命令を実行しキャッシュヒットした場合の各部の動作例を示したブロック図である。

図8において、プロセッサCPU_mは、ロード命令が発行され、該ロード命令で指定された集中共有メモリCSMのアドレスをハードキャッシュ制御部9に出力する。ハードキャッシュ制御部9は、タグをチェックしてプロセッサCPU_m

から入力された集中共有メモリCSMのアドレスのデータがキャッシュメモリ6内にあるか否かをチェックする。

【0042】

キャッシュメモリ6内にプロセッサCPUmから指定されたアドレスのデータがあった場合、ハードキャッシュ制御部9は、キャッシュヒットしたことを示す所定のヒット信号hitをプロセッサCPUmに出力する。次に、ハードキャッシュ制御部9は、タグの対応表に基づいて、プロセッサCPUmから指定されたアドレスのデータをキャッシュメモリ6から読み出してプロセッサCPUmに出力する。このようにして、ハードウェア制御動作が行われる。

【0043】

次に、図9は、ポストアストア(PostStore)実行時における各部の動作例を示したブロック図であり、図9を用いてポストアストア実行時における各部の動作を説明する。なお、図9においても、点線の矢印が制御の流れを、実線の矢印がデータの流れを示しており、ソフトウェア制御時の動作説明に関係しない部分は省略している。

図9において、ポストアストアを実行する場合、まずDTC8は、SAC命令が発行され、該SAC命令で指定されたアドレスにデータを書き込むようキャッシュメモリ6の動作制御を行う。

【0044】

次に、プロセッサCPUmは、ストア命令が発行され、該ストア命令で指定された集中共有メモリCSMのアドレスをDTC8に出力すると共に、キャッシュメモリ6にデータを出し、該データは、上記SAC命令で指定されたアドレスに格納される。DTC8は、キャッシュメモリ6に格納されたデータにおける集中共有メモリCSMのアドレスをタグから求め、ストア命令で指定された集中共有メモリCSMのアドレスと一致するかチェックする。DTC8は、一致しなかった場合は、プロセッサCPUmに対して所定のミス信号missを出力し、一致した場合は、所定のヒット信号hitを出力する。プロセッサCPUmは、ヒット信号hitが入力されると引き続きソフトウェア制御で動作させ、ミス信号missが入力されると、ソフトウェア制御からハードキャッシュ制御部9を使

用するハードウェア制御に切り替える。

【0045】

次に、DTC8は、PS命令が発行され、該PS命令で指定されたキャッシュメモリ6のアドレスに格納されたデータが有効であるか否かをチェックし、データが有効であれば、該データを、指定された集中共有メモリCSMに転送した後、無効化を行う。また、DTC8は、データが無効であれば、所定のミス信号missをプロセッサCPUmに出力し、プロセッサCPUmは、ソフトウェア制御からハードキャッシュ制御部9を使用するハードウェア制御に切り替える。

【0046】

このようにして、DTC8によるポストストアが行われる。なお、上記プレロード及びポストストアにおいて、集中共有メモリCSMにアクセスする必要がない場合、例えばデータがまだプロセッサで使用される場合等では、PL命令及びPS命令を発行しないようにDTC8に対するプログラムコードが生成される。

【0047】

プロセッサCPUm及びDTC8に対する各命令の発行タイミングは、あらかじめコンパイラによってスケジューリングされる。以下、該スケジューリング方法について説明する。

図10は、コンパイラによって行われるプロセッサCPUmに対するアセンブラコード及びDTC8に対するDTCコードの生成過程を示したブロック図である。なお、図10で示した各処理は、通常ワークステーション等のコンピュータによって実行されるコンパイラによって行われる。

【0048】

図10において、まず最初に、C言語又はFORTRAN言語でプログラムされた逐次型のプログラムに対して自動並列化処理が行われ、プログラムを並列性に応じて粗粒度 (Coarse-Grain)、中粒度 (Medium-Grain) 及び細粒度 (Fine-Grain) といった3つの粒度に分けられる。

ここで、自動並列化処理について簡単に説明する。自動並列化処理では、まずプログラムを分岐部分ですべて区切り、分岐のない一連のプログラムに分け、これを粗粒度タスク又はタスクと呼ぶ。

【 0 0 4 9 】

粗粒度並列化では、まず該タスク間に依存がないか調べ、依存がなければ各タスクを並列に処理するようにする。次に、中粒度並列化では、上記タスク内を調べて更に簡単に並列化できるループ、すなわち各イタレーションで計算されるデータに依存がないループであるか否かを調べ、並列化できるループであれば並列化を行う。このような中粒度並列化は、別名ループ並列化ともいう。次に、細粒度並列化では、タスクがループでないか又は並列化できないループであった場合に適用される並列化手法であり、プログラムの各文（例えば $a = b + c$ 等）の間の並列性を抽出する方法である。

【 0 0 5 0 】

次に、上記細粒度並列化処理を行った後のコードで、プロセッサ CPU_m用の分岐のない一連の命令コードである細粒度コードを使用して、アセンブラの CPUコードを生成し、該生成した CPUコードに基づいて DTCコードを生成する。一方、DTC₈をクロックレベルで厳密に動作させるためには、データのロード又はストア等で発生するネットワークパケットの衝突まで予測する必要がある。このことから、ネットワークレベルでのスケジューリングを行うことによって該予測を行い、完全にクロックレベルでの動作を保証した DTCコードを作成する。

【 0 0 5 1 】

次に、上記細粒度コードから生成された CPUコードに基づいて DTCコードを生成する際に実行される、コンパイラの静的予測に基づくソフトウェア制御時の DTC用命令の配置アルゴリズムについて説明する。

まず、コンパイラは、通常のコンパイラ技術によってアセンブラの CPUコードの生成を行い、該生成した CPUコードからデータをロードするロード命令、データを書き込むストア命令を探し、対応する DTC命令、すなわち LAC命令、SAC命令、PL命令及びPS命令を生成する。

【 0 0 5 2 】

次に、コンパイラは、ロード又はストアするデータが必ずキャッシュヒットするという前提条件のもとで、生成した CPUコードをシミュレーションし、各命

令の正確なクロックを算出する。更に、コンパイラは、プロセッサCPUmがロード又はストアするときに確実にキャッシュメモリ6にデータがあるようにDTC用命令であるLAC命令、SAC命令、PL命令及びPS命令の発行タイミングの調整を行う。

【0053】

ここで、コンパイラによって行われるDTC用命令の発行タイミングの決定方法について説明する。

プロセッサCPUmの実行命令には、実行順に0から昇順に実行クロックが決められており、DTC命令列の実行クロックはプロセッサCPUmの命令列の実行クロックを基準とする。したがって、プロセッサCPUmで実行される最初の命令よりも前に実行するDTC命令は、実行クロックがマイナスになる。また、プロセッサCPUm及びDTC8は、各クロックで実行することができる命令は共に1命令のみである。

【0054】

コンパイラは、LAC命令をプロセッサCPUmのロード命令の1クロック前に、SAC命令をプロセッサCPUmのストア命令の1クロック後に必ず実行されるようにDTC命令列に配置し、該配置されたLAC命令及びSAC命令の移動を禁止する。次に、コンパイラは、PL命令を所定のアルゴリズムにしたがってDTC命令列に配置し、この後PS命令を所定のアルゴリズムにしたがってDTC命令列に配置する。

【0055】

この際、DTC8がPL命令又はPS命令を実行する際、集中共有メモリCSMにアクセスするために数十クロックを要するため、コンパイラは、この間、集中共有メモリCSMにアクセスさせる命令をDTC命令列に配置することができない。なお、以下、PL命令実行時における集中共有メモリCSMへのアクセスに要するクロック範囲をロード範囲と呼び、PS命令実行時における集中共有メモリCSMへのアクセスに要するクロック範囲をストア範囲と呼ぶ。

【0056】

一方、コンパイラは、プロセッサCPUmにおけるすべてのロード命令及びス

トア命令に対して、P L 命令及びP S 命令を対応付けするのではなく、データのライフタイムを算出し、最初のプロセッサC P U mによるロード時にP L 命令を、最後のプロセッサC P U mによるストア時にP S 命令を対応付けする。なお、これ以外の場合、キャッシュメモリ6にデータが格納されていることから、コンパイラは、D T C 8によってL A C 命令及びS A C 命令で処理されるようにする。

【 0 0 5 7 】

図 1 1 は、コンパイラによって行われるP L 命令の配置アルゴリズムを示したフローチャートであり、図 1 1 を用いて、P L 命令の配置方法について説明する。なお、図 1 1 の各フローで行われる処理は、特に明記しない限りコンパイラによって行われる。

図 1 1 において、まず最初に、プロセッサC P U mのロード命令実行クロックから、集中共有メモリC S Mへのアクセスに要するクロック数を差し引いたクロックを各P L 命令の仮の実行クロックに設定する（ステップS 1）。次に、実行クロックの最も遅い、すなわち最後に実行されるP L 命令を対象にする（ステップS 2）。

【 0 0 5 8 】

この後、対象となっているP L 命令が、すでにD T C 命令列に配置されているL A C 命令又はS A C 命令と重なっているか否かを調べ（ステップS 3）、重なっている場合（Y E S）は、対象となっているP L 命令をL A C 命令及びS A C 命令と重ならないクロックまで実行クロックを減少させる（ステップS 4）。次に、現在D T C 命令列に配置されているP L 命令が、配置しようとしているP L 命令のロード範囲に重なるか否かを調べ（ステップS 5）、重なる場合（Y E S）は、重なったP L 命令がロード範囲外になるまで対象となっているP L 命令の実行クロックを減少させ（ステップS 6）、この後、ステップS 3に戻る。また、ステップS 3で、重なっていない場合（N O）は、ステップS 5に進む。

【 0 0 5 9 】

次に、ステップS 5で、重なっていない場合（N O）は、対象となっているP L 命令を現在の実行クロックでD T C 命令列に配置する（ステップS 7）。この

後、DTC命令列に設定されていないPL命令があるか否かを調べ（ステップS8）、設定されていないPL命令がある場合（YES）は、DTC命令列に設定されていないPL命令の内、最も実行クロックの遅いPL命令を対象として（ステップS9）、ステップS3に戻る。また、ステップS8で、設定されていないPL命令がない場合（NO）、本フローは終了する。

【0060】

次に、図12は、コンパイラによって行われるPS命令の配置アルゴリズムを示したフローチャートであり、図12を用いて、PS命令の配置方法について説明する。なお、図12の各フローで行われる処理は、特に明記しない限りコンパイラによって行われる。

図12において、まず最初に、プロセッサCPUmのストア命令実行クロックから1クロックを加えたクロックを、各PS命令の仮の実行クロックに設定する（ステップS11）。次に、実行クロックの最も早い、すなわち最初に実行されるPS命令を対象にする（ステップS12）。

【0061】

この後、対象となっているPS命令が、すでにDTC命令列に配置されているLAC命令又はSAC命令と重なっているか否かを調べ（ステップS13）、重なっている場合（YES）は、対象となっているPS命令をLAC命令及びSAC命令と重ならないクロックまで実行クロックを増加させる（ステップS14）。次に、現在DTC命令列に配置されているPL命令のロード範囲及びPS命令のストア範囲に、配置しようとしているPS命令が重なるか否かを調べ（ステップS15）、重なる場合（YES）は、重なったPL命令のロード範囲外又は重なったPS命令のストア範囲外になるまで対象となっているPS命令の実行クロックを増加させ（ステップS16）、この後、ステップS13に戻る。また、ステップS13で、重なっていない場合（NO）は、ステップS15に進む。

【0062】

次に、ステップS15で、重ならない場合（NO）は、現在DTC命令列に配置されているPL命令又はPS命令が、配置しようとしているPS命令のストア範囲に重なるか否かを調べる（ステップS17）。ステップS17で、重なる場

合 (YES) は、すでに DTC 命令列に配置された PL 命令又は PS 命令がストア範囲に重ならなくなるまで、配置しようとしている PS 命令の実行クロックを増加させ (ステップ S 1 8)、ステップ S 1 3 に戻る。

【 0 0 6 3 】

また、ステップ S 1 7 で、重ならない場合 (NO) は、対象となっている PS 命令を現在の実行クロックで DTC 命令列に配置する (ステップ S 1 9)。この後、DTC 命令列に設定されていない PS 命令があるか否かを調べ (ステップ S 2 0)、設定されていない PS 命令がある場合 (YES) は、DTC 命令列に設定されていない PS 命令の内、最も実行クロックの早い PS 命令を対象として (ステップ S 2 1)、ステップ S 1 3 に戻る。また、ステップ S 2 0 で、設定されていない PS 命令がない場合 (NO)、本フローは終了する。

このようにして、コンパイラは、DTC 命令列への各 PL 命令及び各 PS 命令の配置を行う。

【 0 0 6 4 】

図 1 3 は、上記のようなアルゴリズムを用いてコンパイラで生成された DTC 命令列の例を示した図である。なお、図 1 3 において、CPU 命令列における、LW がロード命令を、SW がストア命令をそれぞれ示している。また、集中共有メモリ CSM に対するアクセス (図 1 3 では、メモリアクセスと示す) において、「PreLoad from Mem 32-35 to Cache 0-3」は、集中共有メモリ CSM のアドレス 3 2 ~ 3 5 のデータをキャッシュメモリ 6 のアドレス 0 ~ 3 にプレロードすることを示している。同様に、集中共有メモリ CSM に対するアクセスにおいて、「PostStore from Cache 0-3 to Mem 16-19」は、キャッシュメモリ 6 のアドレス 0 ~ 3 のデータを集中共有メモリ CSM のアドレス 1 6 ~ 1 9 にポストストアすることを示している。

【 0 0 6 5 】

このように、本実施の形態におけるキャッシュメモリシステム装置は、ハードウェア制御で動作する従来のキャッシュメモリシステム装置に、ソフトウェアで制御されるソフトウェア制御モードを追加すると共に、該ソフトウェア制御モード時にデータ転送を管理するプロセッサである DTC 8 を追加し、該 DTC 8 に

対するプログラムの作成を行う際、キャッシュメモリ 6 のライン情報を管理するアルゴリズムをコンパイラに追加した。

【 0 0 6 6 】

このことから、ソフトウェア制御モード時において、DTC 8 は、コンパイラの静的予測が可能な場合、コンパイラが生成したコードにしたがってプロセッサ CPU m が必要とするデータをあらかじめキャッシュメモリ 6 に用意し、コンパイラの静的予測が不可能な場合、プロセッサ CPU m からの情報を受け取ってアドレスを算出することにより動的にデータをキャッシュメモリ 6 に用意することから、キャッシュミスが発生せずにプロセッサに対して無同期実行が可能となる。

【 0 0 6 7 】

なお、上記実施の形態では、複数のプロセッサに対して無同期実行を行うマルチプロセッサシステムに使用する場合を例にして説明したが、本発明は、これに限定するものではなく、シングルプロセッサの場合においても適用することができる。キャッシュヒット率の改善及び処理時間の短縮等を図ることができる。

【 0 0 6 8 】

【発明の効果】

上記の説明から明らかなように、本発明のキャッシュメモリシステム装置によれば、ソフトウェア制御を行うソフトキャッシュ制御部と、ハードウェア制御を行うハードキャッシュ制御部とを備え、プロセッサは、ソフトキャッシュ制御部に対してソフトウェア制御を実行させ、例えばキャッシュミス等が発生してソフトウェア制御が実行不可能になると、ハードキャッシュ制御部に対してハードウェア制御を行わせるようにした。このことから、ある区間内で 1 0 0 % のキャッシュヒット率を得ることができ、複数のプロセッサに対して無同期実行を行うマルチプロセッサシステムを実現することができる。

【 0 0 6 9 】

また、ソフトキャッシュ制御部は、コンパイラの静的予測によって生成されたコードにしたがって所望のデータをキャッシュメモリ内に格納するようにした。このことから、ある区間内で 1 0 0 % のキャッシュヒット率を得ることができる。

【0070】

具体的には、ソフトキャッシュ制御部は、メインメモリの所望のデータを読み出すデータ読み出し命令をプロセッサが実行する前に、該データ読み出し命令で指定されるメインメモリのアドレスのデータを読み出してキャッシュメモリに格納するようにした。このことから、プロセッサによるロード命令等のデータ読み出し命令実行時に100%のキャッシュヒット率を得ることができる。

【0071】

更に、ソフトキャッシュ制御部は、メインメモリの所望のデータを読み出すデータ読み出し命令をプロセッサが実行すると同時に該データ読み出し命令で指定されたメインメモリのアドレスのデータをキャッシュメモリからプロセッサに転送させるようにした。このことから、プロセッサによるロード命令等のデータ読み出し命令実行時における処理時間の短縮を図ることができる。

【0072】

また、具体的には、ソフトキャッシュ制御部は、メインメモリにデータを書き込むデータ書き込み命令をプロセッサが実行する前に、キャッシュメモリに対してプロセッサからのデータを格納するアドレスを指定するようにした。このことから、プロセッサによるストア命令等のデータ書き込み命令実行時におけるキャッシュメモリへのデータ書き込みの失敗をなくすことができる。

【0073】

更に、ソフトキャッシュ制御部は、データ書き込み命令実行時にプロセッサからキャッシュメモリの指定したアドレスに書き込まれたデータを、該データ書き込み命令で指定されたメインメモリのアドレスに書き込むようにした。このことから、プロセッサによるストア命令等のデータ書き込み命令実行時における処理時間の短縮を図ることができる。

【0074】

一方、ハードキャッシュ制御部は、ハードウェア制御時に、複数ウェイのセットアソシエイティブ方式を用いてキャッシュメモリのライン管理を行うようにし、ソフトキャッシュ制御部は、ソフトウェア制御時に、該複数ウェイの内少なく

とも1ウェイをフルアソシエイティブ方式を用いてキャッシュメモリのライン管理を行うようにした。このことから、キャッシュメモリに対するアクセス速度を低下させることなくキャッシュメモリの利用効率を高めることができる。

【0075】

また具体的には、ソフトキャッシュ制御部を、キャッシュメモリに対するデータ転送制御を行う転送制御用プロセッサで構成するようにした。このことから、ソフトウェア制御時に、あらかじめプログラムされたソフトウェアにしたがってキャッシュメモリに対するデータ転送制御を行うことができる。

【図面の簡単な説明】

【図1】 本発明の実施の形態におけるキャッシュメモリシステム装置を使用するマルチプロセッサシステムの例を示した概略のブロック図である。

【図2】 本発明の実施の形態におけるキャッシュメモリシステム装置の例を示した概略のブロック図である。

【図3】 4ウェイ・セットアソシエイティブ方式を用いた場合におけるタグの対応表の例を示した図である。

【図4】 図3で示したタグの対応表のソフトウェア制御時における例を示した図である。

【図5】 図4で示したタグの対応表のハードウェア制御移行時における例を示した図である。

【図6】 図2のキャッシュメモリシステム装置のプレロード実行時における各部の動作例を示したブロック図である。

【図7】 ハードウェア制御時にキャッシュミスした場合の、図2のキャッシュメモリシステム装置における各部の動作例を示したブロック図である。

【図8】 ハードウェア制御時にキャッシュヒットした場合の、図2のキャッシュメモリシステム装置における各部の動作例を示したブロック図である。

【図9】 図2のキャッシュメモリシステム装置のポストアスタ実行時における各部の動作例を示したブロック図である。

【図10】 コンパイラによって行われるプロセッサCPUmに対するアセンブラコード及びDTCコードの生成過程を示したブロック図である。

【図 1 1】 コンパイラによって行われる P L 命令の配置アルゴリズムを示したフローチャートである。

【図 1 2】 コンパイラによって行われる P S 命令の配置アルゴリズムを示したフローチャートである。

【図 1 3】 コンパイラで生成された D T C 命令列の例を示した図である。

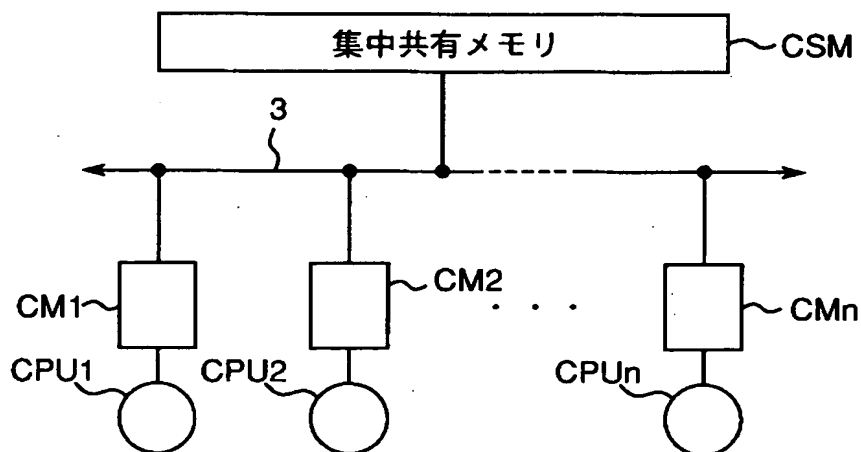
【図 1 4】 従来のキャッシュメモリシステム装置の構成例を示した概略のブロック図である。

【符号の説明】

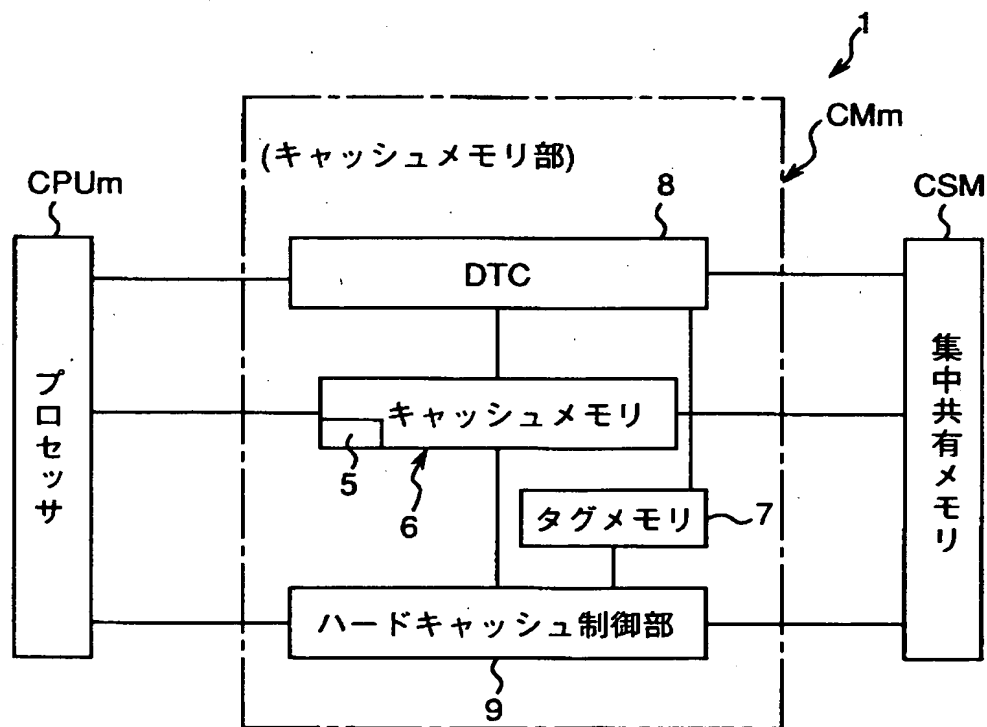
- 1 キャッシュメモリシステム装置
- 5 リードバッファ
- 6 キャッシュメモリ
- 7 タグメモリ
- 8 D T C (ソフトキャッシュ制御部)
- 9 ハードキャッシュ制御部
- C P U 1 ~ C P U n プロセッサ
- C M 1 ~ C M n キャッシュメモリ部
- C S M 集中共有メモリ (メインメモリ)

【書類名】 図面

【図 1】



【図 2】



【図3】

タグ	キー			
	第1ウェイ	第2ウェイ	第3ウェイ	第4ウェイ
0	16	24	—	—
1	01	09	17	25
2	18	10	02	—
3	11	43	19	27
4	20	28	—	—
5	05	13	21	29
6	14	22	—	—
7	07	23	31	—

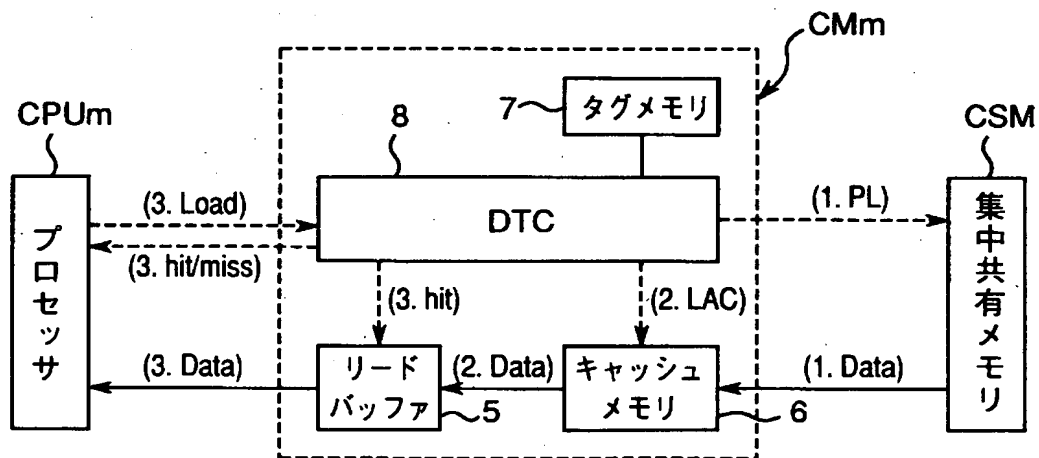
【図4】

タグ	キー			
	第1ウェイ	第2ウェイ	第3ウェイ	第4ウェイ
0	16	24	—	37
1	01	09	17	25
2	18	10	02	45
3	11	43	19	27
4	20	28	—	33
5	05	13	21	29
6	14	22	—	53
7	07	23	31	—

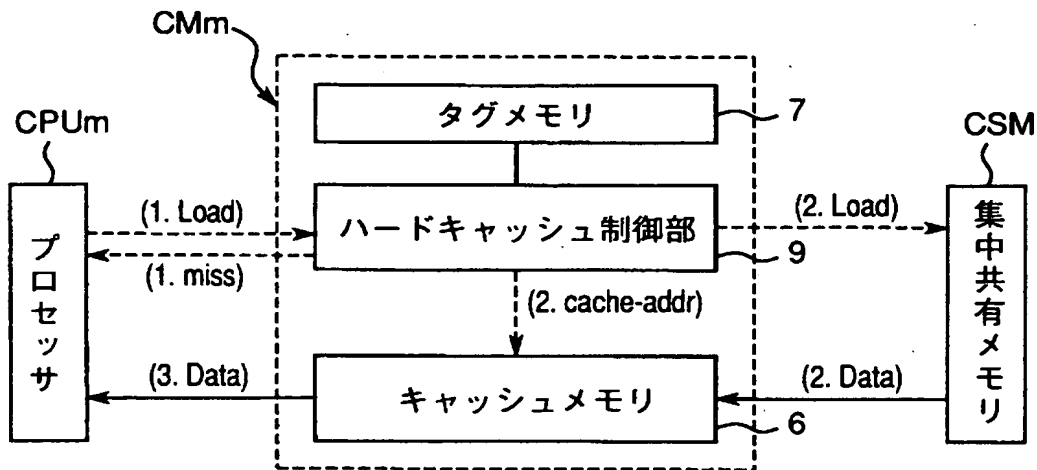
【図5】

タグ	キー			
	第1ウェイ	第2ウェイ	第3ウェイ	第4ウェイ
0	16	24	—	—
1	01	09	17	25
2	18	10	02	—
3	11	43	19	—
4	20	28	—	—
5	05	13	21	—
6	14	22	—	—
7	07	23	31	—

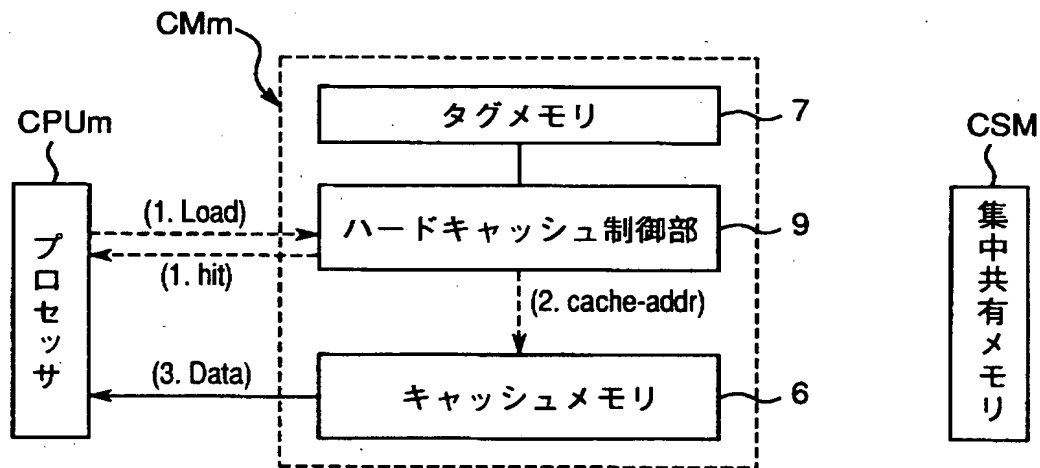
【図6】



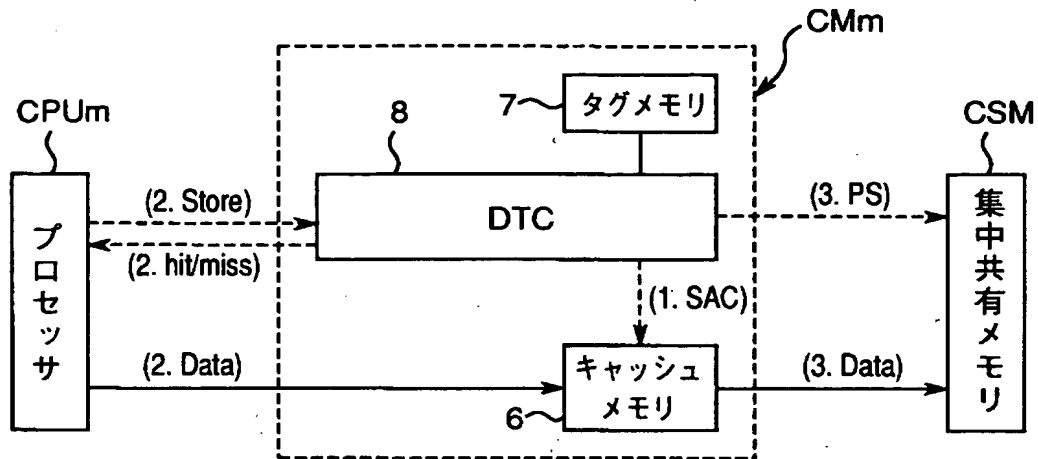
【図 7】



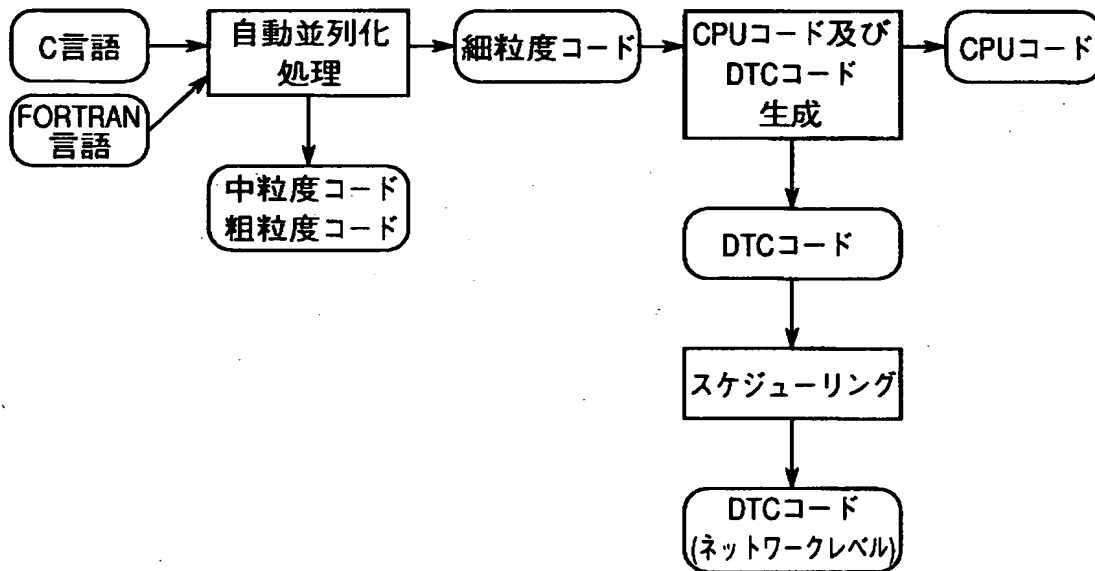
【図 8】



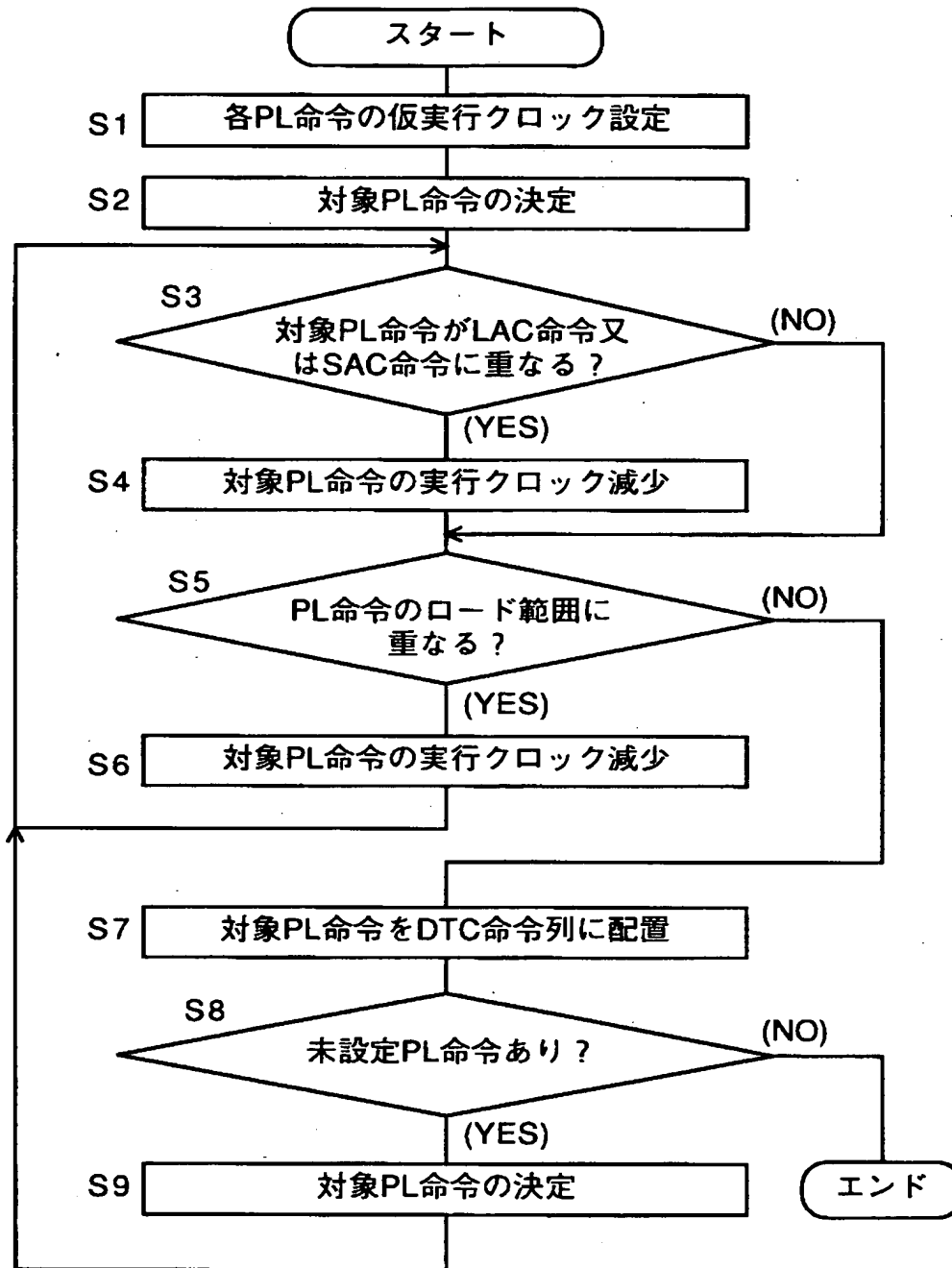
【図 9】



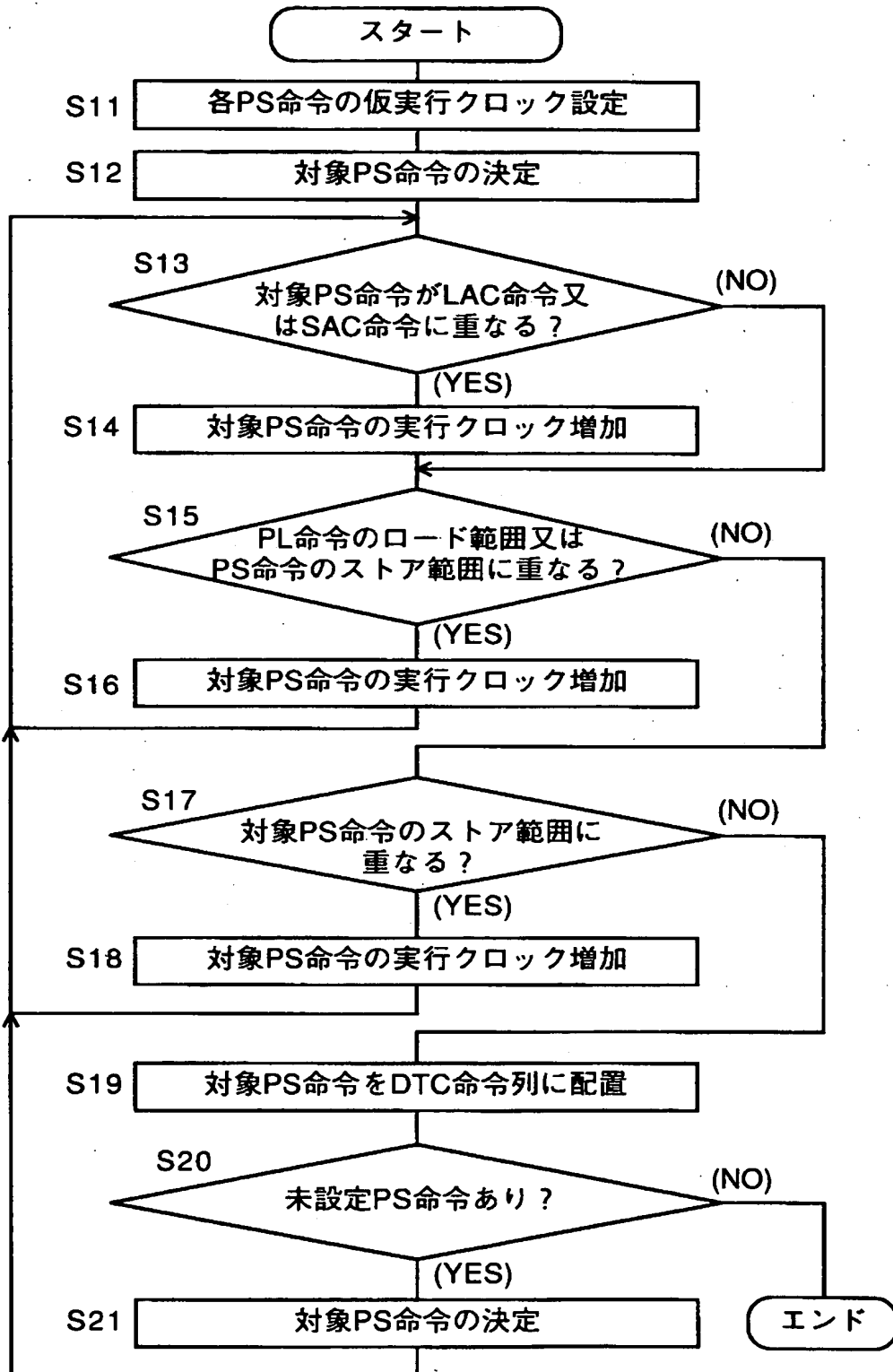
【図 10】



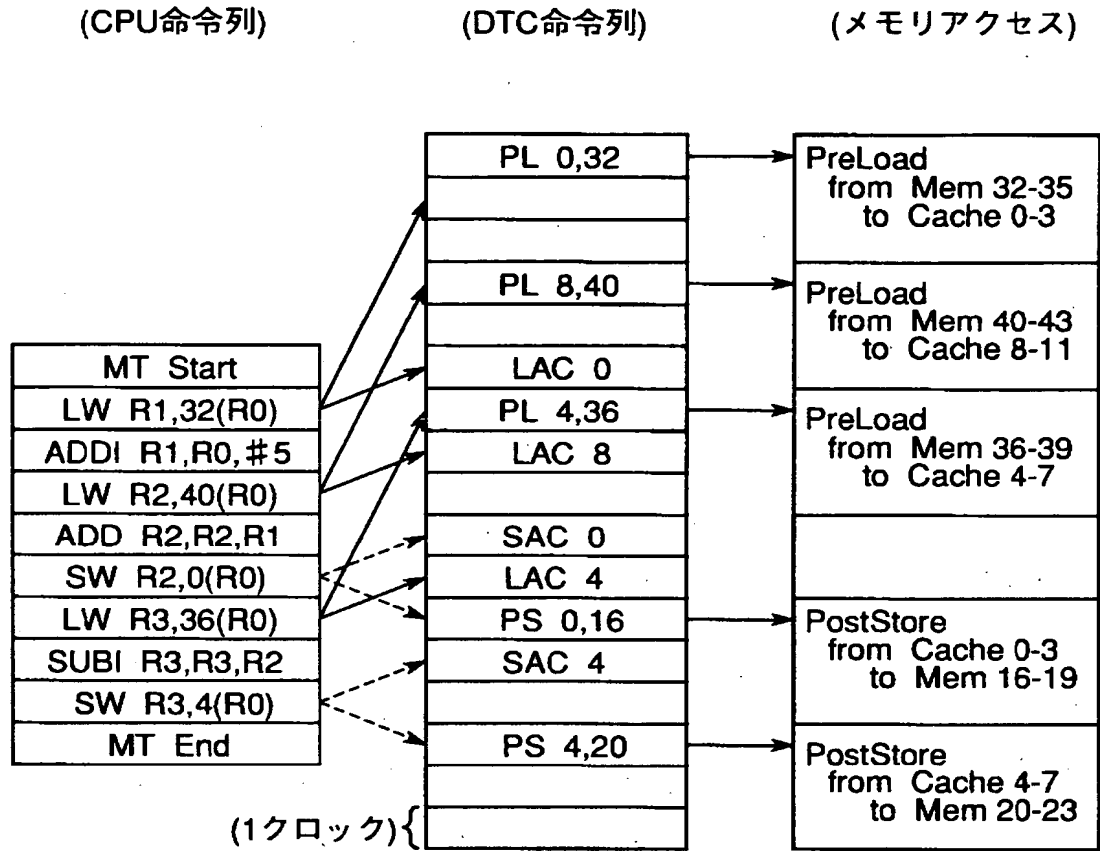
【図 1 1】



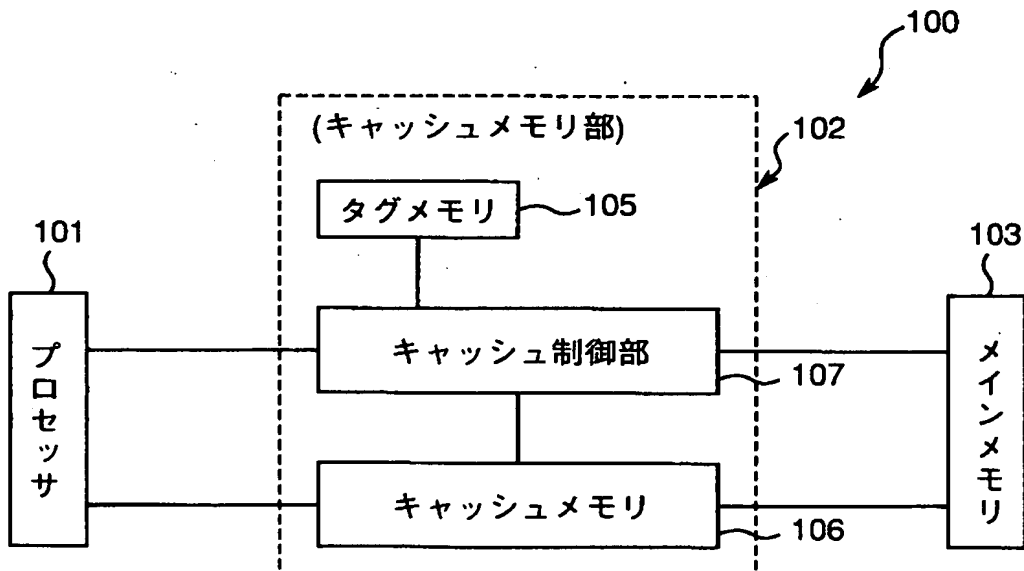
【図 1 2】



【図13】



【図14】



【書類名】 要約書

【要約】

【課題】 キャッシュミスが発生せずにマルチプロセッサシステム等における無同期実行を可能にするキャッシュメモリシステム装置を得る。

【解決手段】 ソフトウェアで制御されるソフトウェア制御モードを追加すると共に、該ソフトウェア制御モード時にデータ転送を管理するプロセッサであるDTC 8を追加し、該DTC 8に対するプログラムの作成を行う際、キャッシュメモリ 6のライン情報を管理するアルゴリズムをコンパイラに追加し、ソフトウェア制御モード時において、DTC 8は、コンパイラの静的予測が可能な場合、コンパイラが生成したコードにしたがってプロセッサCPU mが必要とするデータをあらかじめキャッシュメモリ 6に用意し、コンパイラの静的予測が不可能な場合、プロセッサCPU mからの情報を受け取ってアドレスを算出することにより動的にデータをキャッシュメモリ 6に用意するようにした。

【選択図】 図 2

出 願 人 履 歴 情 報

識別番号 [396023993]

1. 変更年月日 1996年10月28日

[変更理由] 新規登録

住 所 東京都港区新橋6丁目16番10号

氏 名 株式会社半導体理工学研究センター

2. 変更年月日 2001年 3月23日

[変更理由] 住所変更

住 所 神奈川県横浜市港北区新横浜3丁目17番地2 友泉新横浜ビル6階

氏 名 株式会社半導体理工学研究センター